object promotion and/or death-generated updates are performed coincident with a collection interval in which a sampled object is promoted to an older generation or is determined to be unreachable.

Please delete the paragraph beginning on page 18, line 25, and substitute therefor the following paragraph:

4. Counter: A counter can be employed as an arbitrary selection mechanism. For example, in one realization, a counter initialized with some positive value is decremented each time an object is allocated. Selection employs a computationally efficient triggering mechanism. For example, if the carry-bit of the decrement is added into the address of the free pointer, then when the counter underflows, the load from the free pointer will be biased causing a misalignment trap. The trap handler can either perform the sampling directly or patch the allocation site to sample the next allocated object. Such an approach imposes some additional computational load (e.g., 4 extra instructions in the fast-path), but avoids skewing.

Please delete the paragraph beginning on page 28, line 15, and substitute therefor the following paragraph:

[1087] In general, a larger evaluation window will provide more precise information upon which to base a reversal decision. On the other hand, a window that extends far into the allocation history may require excessive data storage and may increase overhead. Accordingly, suitable evaluation windows are, in general, implementation dependent. In any case, if sampling of tenured objects in the evaluation window indicates that a sufficient portion of pretenured objects from an allocation site have died in such a window, then we reverse the allocation decision and patch the allocation-site to once again allocate into the young generation.

In the Claims

Please add the following new claims:

48. (New) The method of claim 1,

wherein at least some of the sampling is performed coincident with death of respective ones of the sampled memory objects.